PATENT ABSTRACTS OF JAPAN

Ref. 2

(11)Publication number : 2001-175619
(43)Date of publication of application . 29.06.2001

(51)Int.Cl. G06F 15/16

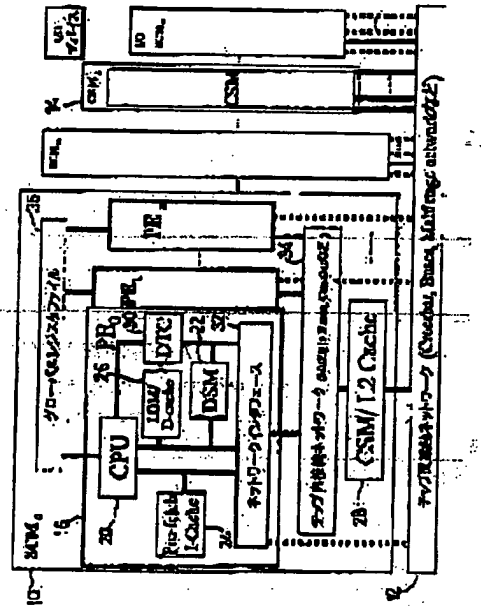
(21)Application number : 11-363702
(22)Date of filing : 22.12.1999

(71)Applicant : UNIV WASEDA
(72)Inventor : KASAHARA HIRONORI
KIMURA KEIJI

11/ 13

(54) SINGLE-CHIP MULTIPROCESSOR

(57)Abstract:
PROBLEM TO BE SOLVED: To achieve performance improvement which is scalable up to an increasing degree of semiconductor integration by improving the price performance ratio with respect to a multiprocessor for parallel processing.
SOLUTION: The single-chip multiprocessor includes processing elements 16 each including a CPU 20, a network interface 32 connected to the CPU, an adjustable prefetch instruction cache 24 connected directly to the CPU and network interface, and a data transfer controller 30 connected directly to the CPU and a concentrated common memory 28 which is connected to the respective processing elements and shared by the processing elements.



(19)日本国特許庁（ＪＰ）　(12) **公開特許公報**（Ａ）　(11)特許出願公開番号
特開2001－175619
（P2001－175619A）
(43)公開日　平成13年６月29日(2001.6.29)

(51)Int.Cl.⁷	識別記号	ＦＩ	ﾃｰﾏｺｰﾄﾞ(参考)
Ｇ０６Ｆ　15/16	６４０	Ｃ０６Ｆ　15/16	６４０Ｂ　５Ｂ０４５

審査請求　未請求　請求項の数４　ＯＬ　（全９頁）

(21)出願番号　特願平11－363702

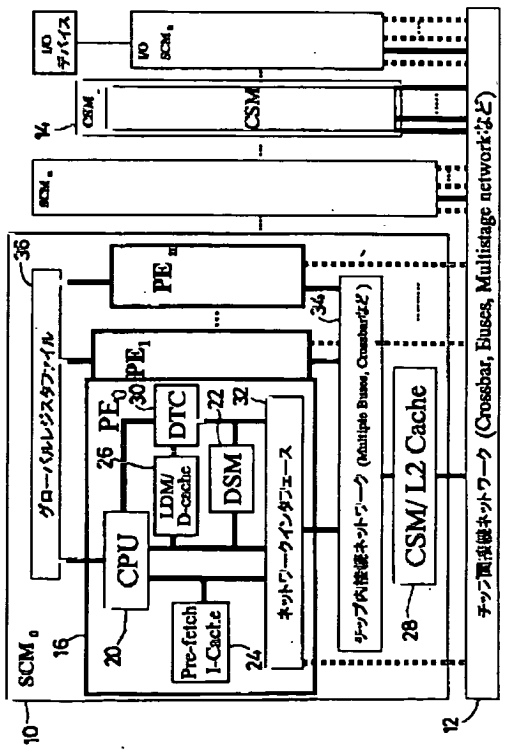
(22)出願日　平成11年12月22日(1999.12.22)

(71)出願人　899000068
学校法人　早稲田大学
東京都新宿区戸塚町１丁目104番地
(72)発明者　笠原　博徳
東京都新宿区大久保３－４－１　早稲田大学理工学部電気電子情報工学科
(72)発明者　木村　啓二
東京都新宿区大久保３－４－１　早稲田大学理工学部電気電子情報工学科
(74)代理人　100099623
弁理士　奥山　尚一　（外２名）

Ｆターム(参考)　5B045 DD01 GG11 KK08

(54)【発明の名称】　シングルチップマルチプロセッサ

(57)【要約】
【課題】　並列処理用のマルチプロセッサにおいて、価格性能比を改善し、高まりつつある半導体集積度にスケーラブルな性能向上を達成する。
【解決手段】　ＣＰＵ２０と、該ＣＰＵに接続しているネットワークインタフェース３２と、該ＣＰＵと該ネットワークインタフェースに直接接続しているアジャスタブルプリフェッチ命令キャッシュ２４と、該ＣＰＵに直接接続しているデータ転送コントローラ３０とを含んでなる複数のプロセッシングエレメント１６と、各プロセッシングエレメントに接続し各プロセッシングエレメントによって共有される集中共有メモリ２８とを含んでなるシングルチップマルチプロセッサ。



【特許請求の範囲】
【請求項１】　ＣＰＵと、該ＣＰＵに接続しているネットワークインタフェースと、該ＣＰＵと該ネットワークインタフェースに直接接続しているアジャスタブルプリフェッチ命令キャッシュと、該ＣＰＵに直接接続しているデータ転送コントローラと、ローカルメモリまたはデータキャッシュとして切り替え可能なメモリと、すべてのプロセッシングエレメントからアクセス可能な分散共有メモリとを含んでなる複数のプロセッシングエレメントと、各プロセッシングエレメントに接続し各プロセッシングエレメントによって共有される集中共有メモリとを含んでなるシングルチップマルチプロセッサ。
【請求項２】　各プロセッシングエレメントに接続しているグローバルレジスタファイルをさらに含む請求項１記載のシングルチップマルチプロセッサ。
【請求項３】　複数の請求項１または２記載のシングルチップマルチプロセッサを含んでなるコンピュータ。
【請求項４】　複数の請求項１または２記載のシングルチップマルチプロセッサと、すべてのシングルマルチプロセッサにより共有されるメモリからなる集中共有メモリチップと、入出力制御用シングルチップマルチプロセッサとを含んでなるコンピュータ。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、複数のＣＰＵを単一のチップに納めたシングルチッププロセッサのアーキテクチャに関し、より具体的には、マルチグレインのコンパイラ協調型シングルチップマルチプロセッサアーキテクチャと、それらを接続した高性能マルチプロセッサシステムアーキテクチャとに関する。
【０００２】
【従来の技術】現在、日本のスーパーコンピュータメーカは世界でもトップのハードウエア技術を有し、現時点でのピーク性能は、数ＴＦＬＯＰＳを越え、２１世紀初頭には数十ＴＦＬＯＰＳ以上のピーク性能を持つマシンが開発されると予想される。しかし、現在のスーパーコンピュータは、ピーク性能の向上とともにプログラムを実行したときの実効性能との差が大きくなっている、すなわち価格性能比が必ずしも優れているとはいえない状況になっている。また、使い勝手としても、ユーザは問題中の並列性を抽出し、ＨＰＦ、ＭＰＩ，ＰＶＭなどの拡張言語あるいはライブラリを用いハードウエアを効果的に使用できるようなプログラムを作成しなければならず、一般のユーザには使い方が難しい、あるいは使いこなせないという問題が生じている。さらに、これらにも起因して、世界の高性能コンピュータの市場を拡大できないということが大きな問題となっている。
【０００３】この価格性能比、使いやすさの問題を解決し、スーパーコンピュータの市場を拡大するためには、ユーザが使い慣れているフォートラン、Ｃ等の逐次型言語で書かれたプログラムを自動的に並列化する自動並列化コンパイラの開発が重要となる。
【０００４】特に、２１世紀初頭の汎用並びに組み込み用マイクロプロセッサ、家庭用サーバからスーパーコンピュータに至るマルチプロセッサシステムの主要アーキテクチャの一つとなると考えられるシングルチップマルチプロセッサについて検討を行うことは重要である。さらに、シングルチップマルチプロセッサについても、従来からある主記憶共有アーキテクチャでは十分な性能と優れた価格性能比は得られない。したがって、プログラム中の命令レベルの並列性、ループ並列性、粗粒度並列性をフルに使用できるマルチグレイン並列処理のように、真に実行すべき命令列からより多くの並列性を抽出し、システムの価格性能比を向上し、誰にでも使えるユーザフレンドリなシステムの構築を可能とする新しい自動並列化コンパイル技術と、それを生かせるようなアーキテクチャの開発が重要である。
【０００５】
【発明が解決しようとする課題】したがって、本発明は、マルチグレイン並列化をサポートするコンパイラ協調型のシングルチップマルチプロセッサおよびそれを結合したハイパフォーマンスマルチプロセッサシステムを提供することを目的とする。
【０００６】
【課題を解決するための手段】本発明は、ＣＰＵと、ネットワークインタフェースと、該ＣＰＵと該ネットワークインターフェースに直接接続しているアジャスタブルプリフェッチ命令キャッシュと、該ＣＰＵに直接接続しているデータ転送コントローラとを含んでなる複数のプロセッシングエレメントと、各プロセッシングエレメントに接続し各プロセッシングエレメントによって共有される集中共有メモリとを含んでなるシングルチップマルチプロセッサを提供する。
【０００７】また、本発明は、そのようなシングルチップマルチプロセッサを複数必要とするメモリ容量あるいはデータ転送性能に応じ、集中共有メモリのみからなる複数の集中共有メモリチップと、さらに入出力制御を行う複数の入出力チップとに接続した構成のマルチプロセッサシステムを提供する。
【０００８】
【発明の実施の形態】本発明はマルチグレイン並列化をサポートするシングルチップマルチプロセッサを提供する。本発明の一実施形態であるシングルチップマルチプロセッサのアーキテクチャを図１に示す。図１においては、複数のプロセッシングエレメント（PE_0，PE_1，．．．，PE_n）を含んでなる複数（ｍ＋１個）のシングルチップマルチプロセッサ（SCM_0、SCM_1、SCM_2、．．．、SCM_m、．．．）１０と、共有メモリのみからなる複数（ｊ＋１個）の集中共有メモリチップ（CSM_0，．．．．，CSM_j）（ただし、ＣＳＭは

要求されるシステム条件によっては１個もなくてもよい）と、入出力制御を行う複数（ｋ＋１個）のシングルチップマルチプロセッサで構成される入出力チップ（Ｉ／Ｏ SCM_0，．．．，Ｉ／Ｏ SCM_k）（ただし、入出力制御に関しては既存技術のプロセッサを用いることもできる）とが、チップ間接続ネットワーク１２によって接続されている。このインタチップ接続ネットワーク１２は、クロスバー、バス、マルチステージネットワークなど既存のネットワーク技術を利用して実現できるものである。

【０００９】図１に示した形態においては、Ｉ／Ｏデバイスは要求される入出力機能に応じてｋ＋１個のＳＣＭで構成される入出力制御チップに接続している構成となっている。さらに、このチップ間接続ネットワーク１２には、システム中の全プロセッシングエレメントにより共有されているメモリのみから構成されるｊ＋１個の集中共有メモリ（ＣＳＭ：centralized shared memory）チップ１４が接続されている。これは、ＳＣＭ１０内にある集中共有メモリを補完する働きをするものである。

【００１０】マルチグレイン並列処理とは、サブルーチン、ループ、基本ブロック間の粗粒度並列性、ループタイプイタレーション間の中粒度並列性（ループ並列性）、ステートメントあるいは命令間の（近）細粒度並列性を階層的に利用する並列処理方式である。この方式により、従来の市販マルチプロセッサシステム用自動並列化コンパイラで用いられていたループ並列化、あるいはスーパースカラ、ＶＬＩＷにおける命令レベル並列化のような局所的で単一粒度の並列化とは異なり、プログラム全域にわたるグローバルかつ複数粒度によるフレキシブルな並列処理が可能となる。

【００１１】［粗粒度タスク並列処理（マクロデータフロー処理）］単一プログラム中のサブルーチン、ループ、基本ブロック間の並列性を利用する粗粒度並列処理は、マクロデータフロー処理とも呼ばれる。ソースとなる例えばフォートランプログラムを、粗粒度タスク（マクロタスク）として、繰り返しブロック（ＲＢ：repetition block)、サブルーチンブロック（ＳＢ：subroutine block)、疑似代入文ブロック（ＢＰＡ：block of pseudo assignment statements)の３種類のマクロタスク（ＭＴ）に分解する。ＲＢは、各階層での最も外側のナチュラルループであり、ＳＢはサブルーチン、ＢＰＡはスケジューリングオーバヘッドあるいは並列性を考慮し融合あるいは分割された基本ブロックである。ここで、ＢＰＡは、基本的には通常の基本ブロックであるが、並列性抽出のために単一の基本ブロックを複数に分割したり、逆に一つのＢＰＡの処理時間が短く、ダイナミックスケジューリング時のオーバヘッドが無視できない場合には、複数のＢＰＡを融合し得一つのＢＰＡを生成する。最外側ループであるＲＢがＤｏａｌｌループであるときは、ループインデクスを分割することにより複数の部分Ｄｏａｌｌループに分割し、分割後の部分Ｄｏａｌｌループを新たにＲＢと定義する。また、サブルーチンＳＢは、可能な限りインライン展開するが、コード長を考慮し効果的にインライン展開ができないサブルーチンはそのままＳＢとして定義する。さらに、ＳＢやＤｏａｌｌ不可能なＲＢの場合、これらの内部の並列性に対し、階層的マクロデータフロー処理を適用する。

【００１２】次に、マクロタスク間の制御フローとデータ依存を解析し、図２のようなマクロフローグラフ（ＭＦＧ）を生成する。ＭＦＧでは、各ノードがマクロタスク（ＭＴ）、点線のエッジが制御フロー、実線のエッジがデータ依存、ノード内の小円が条件分岐文を表している。また、ＭＴ７のループ（ＲＢ）は、内部で階層的にＭＴおよびＭＦＧを定義できることを示している。

【００１３】次に、マクロタスク間制御依存およびデータ依存より各マクロタスクが最も早く実行できる条件（最早実行可能条件）すなわちマクロタスク間の並列性を検出する。この並列性をグラフ表現したのが図３に示すマクロタスクグラフ（ＭＴＧ）である。ＭＴＧでも、ノードはＭＴ、実線のエッジがデータ依存、ノード内の小円が条件分岐文を表す。ただし、点線のエッジは拡張された制御依存を表し、矢印のついたエッジは元のＭＦＧにおける分岐先、実線の円弧はＡＮＤ関係、点線の円弧はＯＲ関係を表している。例えば、ＭＴ６へのエッジは、ＭＴ２中の条件分岐がＭＴ４の方向に分岐するか、ＭＴ３の実行が終了したとき、ＭＴ６が最も早く実行が可能になることを示している。

【００１４】そして、コンパイラは、ＭＴＧ上のＭＴをプロセッサクラスタ（コンパイラあるいはユーザによりソフトウェア的に実現されるプロセッサのグループ）へコンパイル時に割り当てを行う（スタティックスケジューリング）か、実行時に割り当てを行うためのダイナミックスケジューリングコードを、ダイナミックＣＰアルゴリズムを用いて生成し、これをプログラム中に埋め込む。これは、従来のマルチプロセッサのようにＯＳあるいはライブラリに粗粒度タスクの生成、スケジューリングを依頼すると、数千から数万クロックのオーバヘッドが生じてしまう可能性があり、それを避けるためである。このダイナミックなスケジューリング時には、実行時までどのプロセッサでタスクが実行されるか分からないため、タスク間共有データは全プロセッサから等距離に見える集中共有メモリに割り当てられる。

【００１５】また、このスタティックスケジューリングおよびダイナミックスケジューリングコードの生成の時には、各プロセッサ上のローカルメモリあるいは分散共有メモリを有効に使用し、プロセッサ間のデータ転送量を最小化するためのデータローカライゼーション手法も用いられる。

【００１６】データローカライゼーションは、ＭＴＧ上でデータ依存のある複数の異なるループにわたりイタレ

ーション間のデータ依存を解析し（インターループデータ依存解析）、データ転送が最小になるようにループとデータを分割（ループ整合分割）後、それらのループとデータが同一のプロセッサにスケジューリングされるように、コンパイル時にそれらのループを融合するタスク融合方式か、実行時に同一プロセッサへ割り当てられるようにコンパイラが指定するパーシャルスタティックスケジューリングアルゴリズムを用いてダイナミックスケジューリングコードを生成する。このデータローカライゼーション機能を用いて各ローカルメモリの有効利用を行うことができる。

【００１７】またこの際、データローカライゼーションによっても除去できなかったプロセッサ間のデータ転送を、データ転送とマクロタスク処理をオーバーラップして行うことにより、データ転送オーバヘッドを隠蔽しようとするプレロード・ポストストアスケジューリングアルゴリズムも使用される。このスケジューリングの結果に基づいて各プロセッサ上のデータ転送コントローラを利用したデータ転送が実現される。

【００１８】［ループ並列処理（中粒度並列処理）］マルチグレイン並列化では、マクロデータフロー処理によりプロセッサクラスタ（ＰＣ）に割り当てられるループ（ＲＢ）は、そのＲＢがＤｏａｌｌあるいはＤｏａｃｒｏｓｓループの場合、ＰＣ内のプロセッシングエレメント（ＰＥ）に対してイタレーションレベルで並列化処理（分割）される。

【００１９】ループストラクチャリングとしては、以下のような従来の技術をそのまま利用できる。

（ａ）ステートメントの実行順序の変更
（ｂ）ループディストリビューション
（ｃ）ノードスプリッティングスカラエクスパンション
（ｄ）ループインターチェンジ
（ｅ）ループアンローリング
（ｆ）ストリップマイニング
（ｇ）アレイプライベタイゼーション
（ｈ）ユニモジュラー変換（ループリバーサル、パーミュテーション、スキューイング）

【００２０】また、ループ並列化処理が適用できないループに関しては、図４のようにループボディ部を次に述べる（近）細粒度並列処理か、ボディ部を階層的にマクロタスクに分割しマクロデータフロー処理（粗粒度タスク並列処理）を適用する。

【００２１】［（近）細粒度並列処理］ＰＣに割り当てられるＭＴがＢＰＡまたはループ並列化或いは階層的にマクロデータフロー処理を適用できないＲＢ等の場合には、ＢＰＡ内部のステートメント或いは命令を近細粒度タスクとしてＰＣ内プロセッサで並列処理する。

【００２２】マルチプロセッサシステム或いはシングルチップマルチプロセッサ上での近細粒度並列処理では、プロセッサ間の負荷バランスだけでなくプロセッサ間データ転送をも最少にするようにタスクをプロセッサにスケジューリングしなければ、効率よい並列処理は実現できない。さらに、この近細粒度並列処理で要求されるスケジューリングでは、図４のタスクグラフに示すように、タスク間にはデータ依存による実行順序の制約があるため強ＮＰ完全な非常に難しいスケジューリング問題となる。このグラフは、無サイクル有向グラフである。図中、各タスクは各ノードに対応している。ノード内の数字はタスク番号ｉを表し、ノードの脇の数字はプロセッシングエレメント上でのタスク処理時間t_iを表す。また、ノードN_iからN_jに向けて引かれたエッジは、タスクT_iがT_jに先行するという半順序制約を表している。タスク間のデータ転送時間も考慮する場合、各々のエッジは一般に可変な重みを持つ。タスクT_iとT_jが異なるプロセッシングエレメントへ割り当てられた場合、この重みt_{ij}がデータ転送時間となる。図４においては、データ転送および同期に要する時間を９クロックと仮定している。逆にこれらのタスクが同一プロセッシングエレメントに割り当てられた場合、重みt_{ij}は０となる。

【００２３】このようにして生成されたタスクグラフを各プロセッサにスタティックにスケジューリングする。この際、スケジューリングアルゴリズムとして、データ転送オーバヘッドを考慮し実行時間を最小化するヒューリスティックアルゴリズム、例えばＣＰ／ＤＴ／ＭＩＳＦ法、ＣＰ／ＥＴＦ／ＭＩＳＦ法、ＥＴＦ／ＣＰ法、あるいはＤＴ／ＣＰ法の４手法を自動的に適用し最良のスケジュールを選ぶことができる。また、このようにタスクをスタティックにプロセッサに割り当てることにより、ＢＰＡ内で用いられるデータのローカルメモリ、分散共有メモリ、レジスタへの配置等、データのメモリへの最適化やデータ転送・同期オーバヘッドの最小化といった各種の最適化が可能になる。

【００２４】スケジューリング後、コンパイラはプロセッシングエレメントに割り当てられたタスクの命令列を順番に並べ、データ転送命令や同期命令を必要な箇所に挿入することにより、各プロセッサ用のマシンコードを生成する。近細粒度タスク間の同期にはバージョンナンバー法を用い、同期フラグの受信は受信側プロセッシングエレメントのビジーウェイトによって行われる。ここで、データ転送および同期フラグのセットは、送信側のプロセッサが受信側のプロセッサ上の分散共有メモリに直接書き込むことにより低オーバヘッドで行うことができる。

【００２５】マシンコード生成時、コンパイラはスタティックスケジューリングの情報を用いたコード最適化を行うことができる。例えば、同一データを使用する異なるタスクが同一プロセッシングエレメントに割り当てられたとき、レジスタを介してそのデータを受け渡しすることができる。また、同期のオーバヘッドを最小化する

ため、タスクの割り当て状況や実行順序から、冗長な同期を除去することもできる。特に、シングルチップマルチプロセッサでは、コード生成時に厳密なコード実行スケジューリングを行うことにより、実行時のデータ転送タイミングを含めたすべての命令実行をコンパイラが制御し、すべての同期コードを除去して並列実行を可能とする無同期並列化のような究極的な最適化も行える。

【００２６】上述のようなマルチグレイン並列処理をマルチプロセッサシステム上で実現するため、一例として、シングルチップマルチプロセッサ（ＳＣＭ）１０は図１に示すようなアーキテクチャを有する。

【００２７】図１において示したアーキテクチャにおいては、ＣＰＵ２０に加えて、分散共有メモリ（ＤＳＭ：distributed shared memory）２２とアジャスタブルプリフェッチ命令キャッシュ２４が各ＳＣＭ１０に設けられている。ここで用いられるＣＰＵ２０は、特に限定されず、整数演算や浮動小数点演算が可能なものであればよい。例えば、ロード／ストアアーキテクチャのシンプルなシングルイッシューＲＩＳＣアーキテクチャのＣＰＵを用いることができるほか、スーパースカラプロセッサ、ＶＬＩＷプロセッサなども用いることができる。分散共有メモリ２２は、デュアルポートメモリで構成されており、他のプロセッシングエレメントからも直接リード／ライトができるようになっており、上に説明した近細粒度タスク間のデータ転送に使用する。

【００２８】アジャスタブルプリフェッチ命令キャッシュ２４は、コンパイラあるいはユーザからの指示で、将来実行すべき命令をメモリあるいは低レベルキャッシュからプリフェッチするものである。このアジャスタブルプリフェッチ命令キャッシュ２４は、複数ウェイのセットアソシアティブキャッシュにおいて、コンパイラ等のソフトから指示される、あるいはハードにより事前に決められたウェイに、将来実行されるライン（命令列）をフェッチできるようにするものである。その際、フェッチの単位としては、複数ラインの連続転送指示も行える。アジャスタブルプリフェッチ命令キャッシュ２４は、命令キャッシュへのミスヒットを最小化させ、命令実行の高速化を可能にするコンパイラによる調整および制御を可能にするキャッシュシステムである。

【００２９】すなわち、このアジャスタブルプリフェッチ命令キャッシュ２４は、すべてのプログラム（命令列）がメモリサイズより小さいことを仮定しているローカルプログラムメモリとは異なり、大きなプログラムにも対応することができ、プログラムの特徴に応じ、プリフェッチをしない通常のキャッシュとしても使用できるし、逆にすべてコンパイラ制御によるプリフェッチキャッシュとして使え、ミスヒットのない（ノーミスヒット）キャッシュとして使用できるものである。

【００３０】このようなアジャスタブルプリフェッチ命令キャッシュの構造の一例を図５に示す。図５に示されたｎウェイのセットアソシエイティブキャッシュにおいては、コンパイラあるいはユーザがプログラムに応じて指定するｊウェイをプリフェッチ（事前読み出し）するエリアとして使用できるものである。コンパイラにより挿入されたプリフェッチ命令（ラインごとではなく複数ラインのプリフェッチも可能）により、命令実行の前に必要な命令が命令キャッシュ上に存在することを可能とし、高速化が実現できる。プロセッシングエレメントは、ｎウェイすべてを通常のキャッシュと同様に読み出すことができる。ラインのリプレースは通常のＬＲＵ（least recently used）法で行われる。そして、各セット（集合）中のウェイには、通常、自由に転送されたラインを格納できるが、プリフェッチ用に指定されたウェイにはプリフェッチ命令によってＣＳＭから転送されたラインのみ格納される。それ以外のウェイは通常のキャッシュと同様にラインを割り当てられる。プリフェッチキャッシュコントローラは、コンパイラからの指示により、命令をＣＳＭからプリフェッチする。このときの転送の単位は、１ラインから複数ラインである。コンパイラがｊウェイ分のプリフェッチエリアを指定し、それ以外の（ｎ－ｊ）ウェイ分のエリアは通常のキャッシュとして使用される。

【００３１】さらに、図１のアーキテクチャにおいては、ローカルデータメモリ（ＬＤＭ）２６が設けられている。このローカルデータメモリ２６は、各プロセッシングエレメント１６内だけでアクセスできるメモリであり、データローカライゼーション技術などにより、各プロセッシングエレメント１６に割り当てられたタスク間で使用されるローカルデータを保持するために使用される。また、このローカルデータメモリ２６は、対象とするアプリケーションプログラムに対しコンパイラあるいはユーザがデータのローカルメモリへの分割配置が可能な場合には、ローカルメモリとして使用され、ローカルメモリを有効に使用できない場合には、レベル１キャッシュ（Ｄキャッシュ）に切り替えて使用できるようにすることが好ましい。また、ゲーム機等のリアルタイム応用に専ら用いられるような場合には、ローカルメモリだけとして設計することも可能である。基本的に各プロセッシングエレメント内で使用されるメモリであるため、共有メモリに比べチップ面積を消費しないので、相対的に大きな容量をとれるものである。

【００３２】粗粒度並列処理では、条件分岐に対処するためにダイナミックスケジューリングが使用される。この場合、マクロタスクがどのプロセッサで実行されるかは、コンパイル時には分からない。したがって、ダイナミックにスケジューリングされるマクロタスク間の共有データは、集中共有メモリ（ＣＳＭ：centralized shared memory）に配置できることが好ましい。そのため、本実施形態においては、各プロセッシングエレメント１６が共有するデータを格納する集中共有メモリ２８を各Ｓ

CM内に設けるほか、さらに、チップ間接続ネットワーク１２につながれた集中共有メモリ１４を設けている。このチップ内の集中共有メモリ２８は、チップ１０内のすべてのプロセッシングエレメント１６から、そして複数チップの構成では他のチップ上のプロセッシングエレメントからも共有されるデータを保存するメモリである。チップ外の集中共有メモリ１４も同様に各プロセッシングエレメントにより共有されるメモリである。したがって、実際の設計上、集中共有メモリ２８、１４は、物理的に各チップに分散されているが、論理的にはどのプロセッシングエレメントからも等しく共有することができるものである。すべてのプロセッシングエレメントから等距離に見えるようにインプリメントすることもできるし、自チップ内のプロセッシングエレメントからは近く見えるようにインプリメントすることをも可能である。

【００３３】単一のＳＣＭチップからなるシステムでは、チップ内のプロセッシングエレメント（ＰＥ）１６間で共有される等距離の共有メモリとしてこの集中共有メモリ２８を用いることができる。また、コンパイラの最適化が困難である場合には、Ｌ２キャッシュとして使用することができる。このメモリ２８，１４には、ダイナミックタスクスケジューリング時にタスク間で共有されるデータを主に格納する。また、別のチップとなった集中共有メモリ１４は、ＳＣＭチップ１０内の集中共有メモリ２８の容量が足りない場合、必要に応じて、メモリのみからなる大容量集中共有メモリチップを任意の数接続することができる。

【００３４】また、粒度によらずスタティックスケジューリングが適用できる場合には、あるマクロタスクが定義する共有データをどのプロセッサが必要とするかはコンパイル時に分かるため、生産側のプロセッサが消費側のプロセッサの分散共有メモリにデータと同期用のフラグを直接書き込めることが好ましい。

【００３５】データ転送コントローラ（ＤＴＣ）３０は、コンパイラあるいはユーザの指示により自プロセッシングエレメント上のＤＳＭ２２や、自あるいは他のＳＣＭ１０内のＣＳＭ２８、あるいは他のプロセッシングエレメント上のＤＳＭとの間でデータ転送を行う。複数のＳＣＭからなる構成を採用する場合には、他のＳＣＭ上のＣＳＭやＤＳＭとの間でのデータ転送、あるいは、独立したＣＳＭとの間でのデータ転送を行う。

【００３６】図１におけるローカルデータメモリ２６とデータ転送コントローラ３０との間の点線は、用途に応じて、データ転送コントローラ３０がローカルデータメモリ（Ｄキャッシュ）２６にアクセスできる構成をとってもよいことを表している。このような場合、ローカルデータメモリ２６を介してＣＰＵ２０が転送指示をデータ転送コントローラ３０に与えたり、転送終了のチェックを行う構成をとることができる。

【００３７】データ転送コントローラ３０へのデータ転送の指示は、ローカルデータメモリ２６、ＤＳＭ２２、あるいは専用のバッファ（図示しない）を介して行い、データ転送コントローラ３０からＣＰＵ２０へのデータ転送終了の報告は、ローカルメモリ、ＤＳＭあるいは専用のバッファを介して行う。このとき、どれを使うかはプロセッサの用途に応じプロセッサ設計時に決めるかあるいはハード的に複数の方法を用意し、プログラムの特性に応じコンパイラあるいはユーザがソフト的に使い分けられるようにする。

【００３８】データ転送コントローラ３０へのデータ転送指示（例えば何番地から内バイトのデータをどこにストアし、またロードするか、データ転送のモード（連続データ転送、ストライド、ストライド・ストライド転送など）など）は、コンパイラが、データ転送命令をメモリあるいは専用バッファに格納しておき、実行時にはどのデータ転送命令を実行するかの指示のみを出すようにして、データ転送コントローラ２０の駆動のためのオーバヘッドを削減することが好ましい。

【００３９】各ＳＣＭチップ１０内のプロセッシングエレメント１６の間の接続は、各プロセッシングエレメントに設けられたネットワークインタフェース３２を介して、チップ内接続ネットワーク（マルチバス、クロスバーなどからなる）３４によって達成されており、このチップ内接続ネットワーク３４を介して、プロセッシングエレメントが共通の集中共有メモリ２８に接続される。集中共有メモリ２８は、チップの外にあるチップ間接続ネットワーク１２に接続している。このチップ間接続ネットワークは、クロスバーネットワークあるいはバス（複数バスも含む）が特に好ましいが、多段結合網等でもかまわず、予算、ＳＣＭの数、アプリケーションの特性に応じて選ぶことができるものである。また、このチップ内接続ネットワーク３４を介さずに、外部のチップ間接続ネットワーク１２とネットワークインタフェース３２を接続することも可能であり、このような構成は、システム中の全プロセッシングエレメントが平等に各チップ上に分散された集中共有メモリ、分散共有メモリにアクセスすることを可能にするほか、チップ間でのデータ転送が多い場合には、この直結パスを設けることにより、システム全体のデータ転送能力を大幅に高めることができる。

【００４０】グローバルレジスタファイル３６は、マルチポートレジスタであり、チップ内のプロセッシングエレメントにより共有されるレジスタである。たとえば、近細粒度タスク（分散共有メモリを用いた場合など）のデータ転送および同期に使用することができる。このグローバルレジスタファイルは、プロセッサの用途に応じて、省略することも可能なものである。

【００４１】図１において、点線は、通信線を必要に応じて用意できることを意味しており、コストあるいはピ

ン数などを考えて不必要あるいは困難な場合には、点線の接続はなくても動作することを示すものである。

【００４２】以上のように、特定の実施の形態に基づいて本発明を説明してきたが、本発明の技術的範囲はこのような実施の形態に限定されるものではなく、当業者にとって容易な種々の変形を含むものである。

【００４３】

【発明の効果】上述のように、本発明のシングルチップマルチプロセッサによれば、価格性能比を改善し、高まりつつある半導体集積度にスケーラブルな性能向上が可能である。また、本発明は、このようなシングルチップマルチプロセッサを複数含むシステムをも提供するが、そのようなシステムは、より一層の高速処理を可能にするものである。

【図面の簡単な説明】

【図１】本発明の１実施形態であるマルチグレイン並列処理用システムを示すブロックダイアグラムである。

【図２】本発明において用いることができるコンパイラにおける粗粒度並列処理のためのマクロフローグラフの一例を示すグラフである。

【図３】本発明において用いることができるコンパイラにおける粗粒度並列処理のためのマクロタスクグラフの一例を示すグラフである。

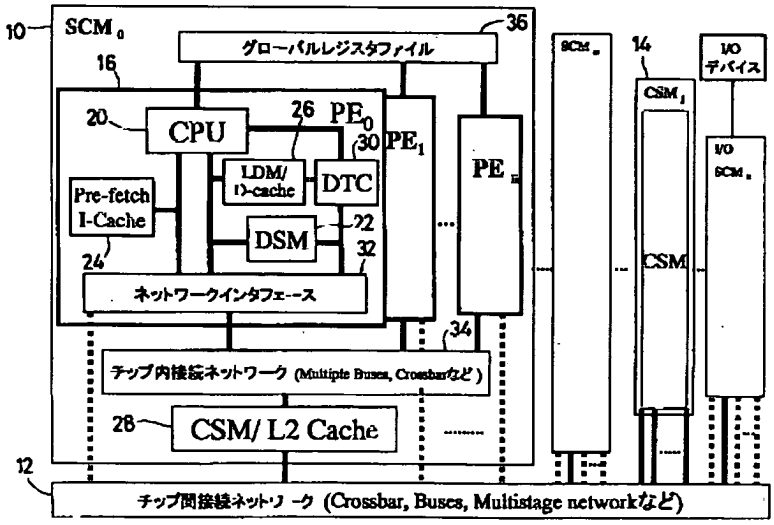
【図４】本発明において用いることができるコンパイラにおける近細粒度並列処理のための近細粒度タスクグラフの一例を示すグラフである。

【図５】本発明において用いることができるアジャスタブルプリフェッチ命令キャッシュの構成を示すブロックダイアグラムである。

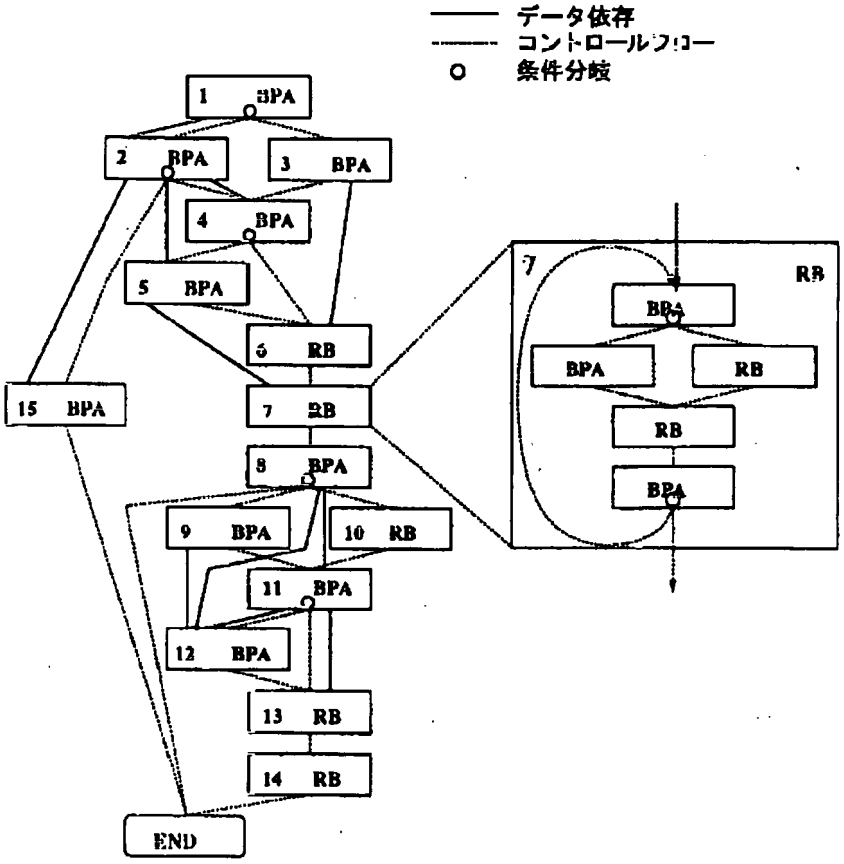
【符号の説明】

１０　シングルチップマルチプロセッサ
１２　チップ間接続ネットワーク
１４　集中共有メモリ（チップ）
１６　プロセッシングエレメント
２０　ＣＰＵ
２２　分散共有メモリ
２４　アジャスタブルプリフェッチ命令キャッシュ
２６　ローカルデータメモリ
２８　集中共有メモリ
３０　データ転送コントローラ
３２　ネットワークインタフェース
３４　チップ内接続ネットワーク

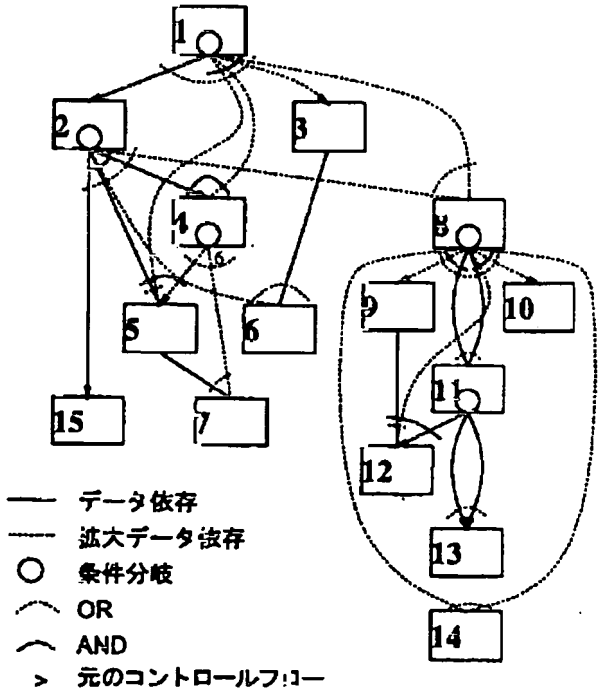
【図１】



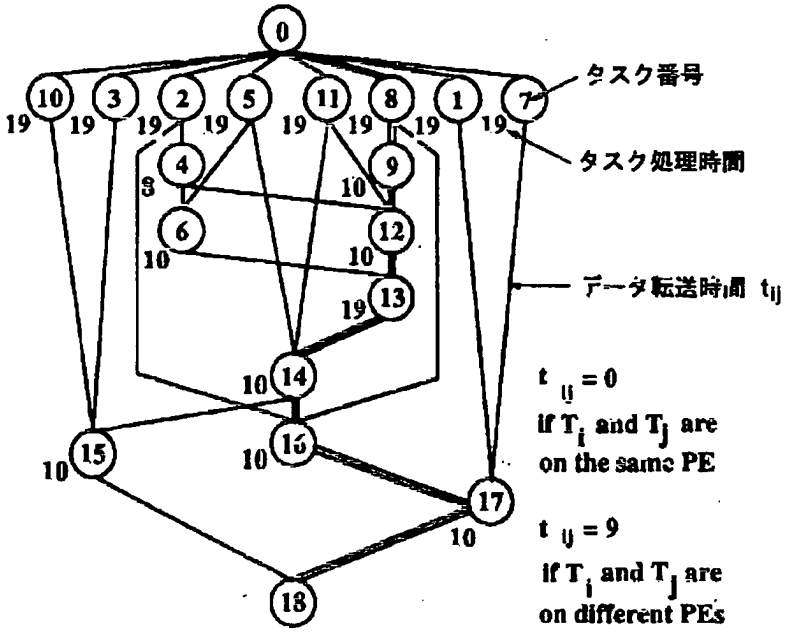
【図2】


データ依存
コントロールフロー
条件分岐
1 BPA
2 BPA
3 BPA
4 BPA
5 BPA
6 RB
7 RB
8 BPA
9 BPA
10 RB
11 BPA
12 BPA
13 RB
14 RB
15 BPA
END
RB
BPA
BPA
RB
RB
BPA

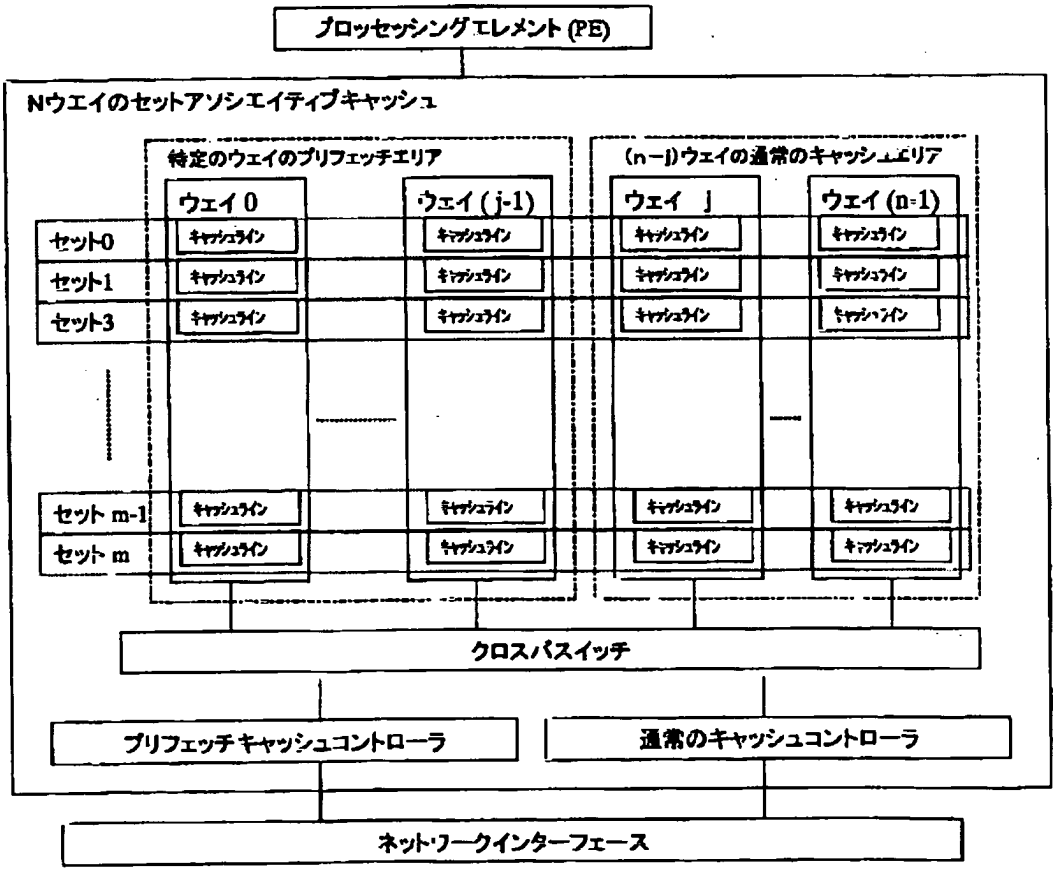
【図3】


データ依存
拡大データ依存
条件分岐
OR
AND
元のコントロールフロー

【図4】


タスク番号
タスク処理時間
データ転送時間 t_{ij}
$t_{ij} = 0$
if T_i and T_j are
on the same PE
$t_{ij} = 9$
if T_i and T_j are
on different PEs

【図5】


プロッセッシングエレメント (PE)
Nウエイのセットアソシエイティブキャッシュ
特定のウエイのプリフェッチエリア
(n−j)ウエイの通常のキャッシュエリア
ウェイ 0
ウェイ (j-1)
ウェイ j
ウェイ (n-1)
セット0
セット1
セット3
セット m-1
セット m
キャッシュライン
クロスバスイッチ
プリフェッチキャッシュコントローラ
通常のキャッシュコントローラ
ネットワークインターフェース